VideoFlexTok: Flexible-Length Coarse-to-Fine Video Tokenization

Andrei Atanov [*1,2] **Jesse Allardice** [*1] **Roman Bachmann** [2] **Oğuzhan Fatih Kar** [2]
Peter Fu [1] **David Griffiths** [1] **Devon Hjelm** [1] **Afshin Dehghan** [1] **Amir Zamir** [2]

[1]Apple [2]Swiss Federal Institute of Technology Lausanne (EPFL)

https://videoflextok.epfl.ch

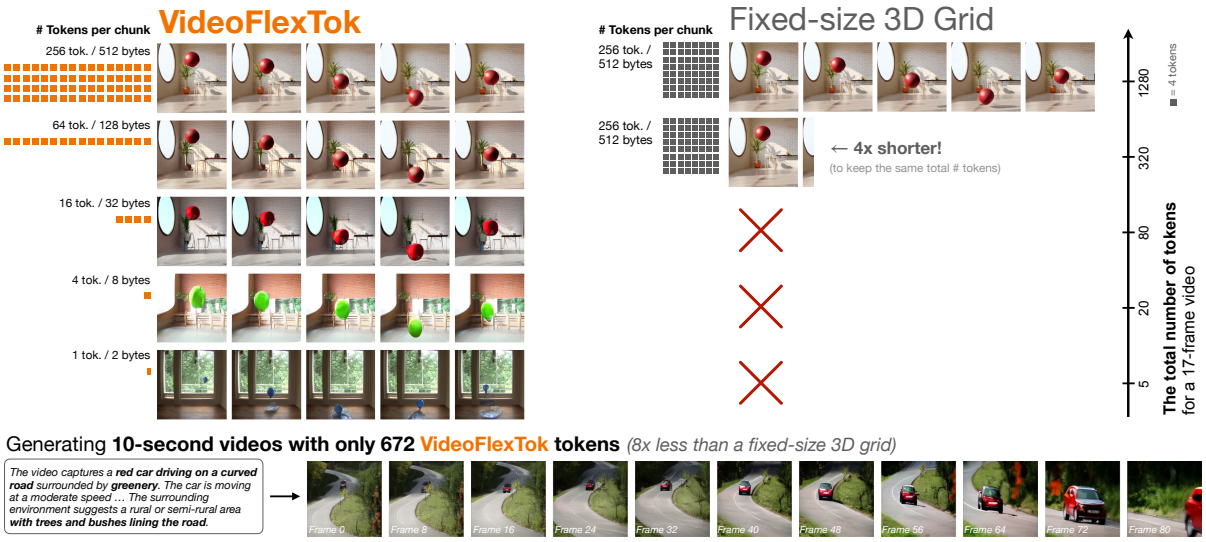

Figure 1. **`VideoFlexTok` represents videos with a flexible-length coarse-to-fine sequence of tokens.** *Top:* Compared to the common 3D grid tokenizers, which can adjust the token sequence length only by reducing the video length or resolution, `VideoFlexTok` can represent the same-length video with a *varying number of tokens corresponding to different levels of detail* – with just a few tokens *emergently* capturing abstract information, such as semantics and motion. *Bottom:* This property enables efficiency, demonstrated here by training a text-to-video model to generate 10-second 81-frame videos using just 672 tokens; 8× fewer than the 5376 required by a comparable 3D grid tokenizer (Agarwal, 2025; Tang et al., 2024; Yu et al., 2023b).

Abstract

Visual tokenizers map high-dimensional raw pixels into a compressed representation for downstream modeling. Beyond compression, tokenizers dictate what information is preserved and how it is organized. A *de facto* standard approach to video tokenization is to represent a video as a spatiotemporal 3D grid of tokens, each capturing the corresponding local information in the original signal. This requires the downstream model that consumes the tokens, e.g., a text-to-video model, to learn to predict all low-level details "pixel-by-pixel" irrespective of the video's inherent complexity, leading to high learning complexity.

We present `VideoFlexTok`, which represents videos with a *variable-length sequence of tokens structured in a coarse-to-fine manner* – where the first tokens (emergently) capture abstract information, such as semantics and motion, and later tokens add fine-grained details. The generative flow decoder enables realistic video reconstructions from any token count. This representation structure allows adapting the token count according to downstream needs and encoding videos longer than the baselines with the same budget.

We evaluate `VideoFlexTok` on class- and text-to-video generative tasks and show that it leads to more efficient training compared to 3D grid tokens, e.g., achieving comparable generation quality (gFVD and ViCLIP Score) with a 5x smaller model (1.1B vs 5.2B). Finally, we demonstrate how `VideoFlexTok` can enable long video generation without prohibitive computational cost by training a text-to-video model on 10-second 81-frame videos with only 672 tokens, 8x fewer than a comparable 3D grid tokenizer.

Correspondence to: Andrei Atanov <andrei.atanov@epfl.ch>.

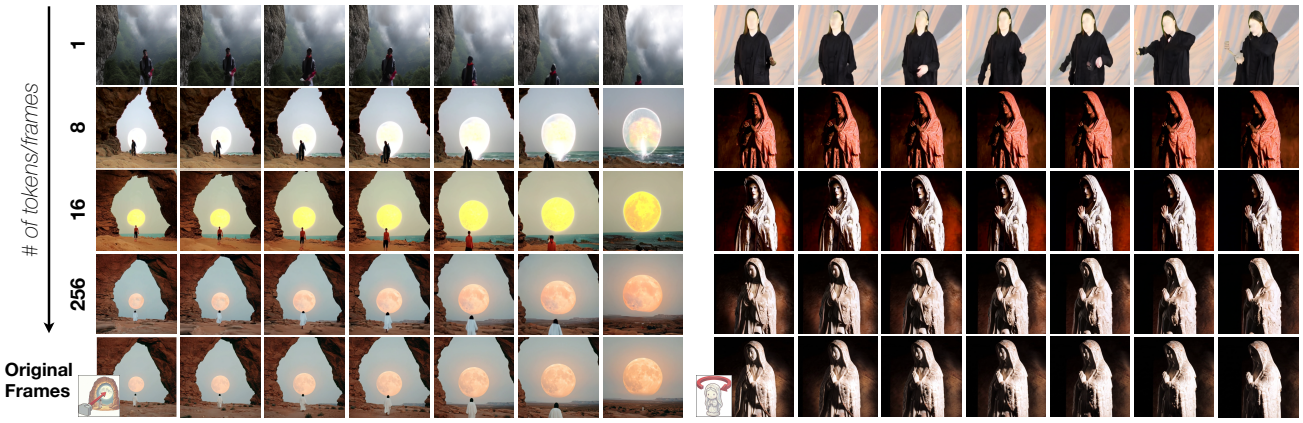

Figure 2. **`VideoFlexTok` reconstructions from a variable number of tokens.** We find that just a few `VideoFlexTok` tokens capture information such as the semantic identities (e.g., a woman in the right example), scene geometry (the *"arch"*), camera motion (moving forward), and object motion (rotation).

1. Introduction

Video modeling[1] is computationally expensive, primarily triggered by the high dimensionality of the raw pixel signal that increases with the length of the video (OpenAI, 2025; 2024; DeepMind, 2024; Wan et al., 2025; HaCohen et al., 2024; Kong et al., 2024; Yang et al., 2024; Kondratyuk et al., 2023). Visual tokenization aims to alleviate this problem by compressing the raw visual signal into a lower-dimensional latent space (Rombach et al., 2022; Van Den Oord et al., 2017; Esser et al., 2021).

Beyond just compression, however, tokenizers also define *what* information is preserved and *how* it is structured within the representation. These are important properties that influence the downstream performance (Ramanujan et al., 2025; Zheng et al., 2025). Most common video tokenizers structure their representations as a fixed-size spatiotemporal 3D grid of tokens, each corresponding to some local information in the original signal (Agarwal, 2025; Tang et al., 2024; Yu et al., 2023b). As a result, any video is represented by the same number of tokens regardless of the complexity of its content. In addition, most tokenizers aim for accurate reconstruction, thereby prioritizing the preservation of pixel-level details in their representations. Therefore, a downstream, e.g., text-to-video, generative model that consumes the tokens needs to learn to simultaneously predict low-level details as well as the more abstract structure, such as the overall semantics and motion. This leads to unnecessary computational cost.

This work presents `VideoFlexTok`, a video tokenizer that *represents videos with a flexible-length sequence of tokens ordered in a coarse-to-fine manner*. The first tokens *emergently* encode the most salient semantic, geometric, and motion information, and later tokens add fine-grained details. `VideoFlexTok`'s decoder is a generative rectified flow model that can produce realistic videos given any number of tokens. This representation structure enables *adaptively and drastically*[2] reducing the signal dimensionality while preserving useful abstract information (see Figures 1 and 2).

We evaluate `VideoFlexTok` on class-to-video and text-to-video downstream tasks. We show that, compared to *de facto* standard fixed-size 3D grid tokenizers, using `VideoFlexTok` results in much lower downstream computational cost. For example, we find that one can achieve a comparable level of performance with an order of magnitude less compute (e.g., using a 10× smaller, 0.4B vs. 3.6B, model.) Finally, we demonstrate how these properties can enable modeling videos of longer duration without extensively increasing the computational cost. Specifically, we train a text-video model directly on 10-second videos represented with only 672 tokens, 8× fewer than standard 3D grid tokenizers, yet capturing most useful semantic and motion information (see Figure 1).

2. Related Work

VAE-based grid tokenization. VAE and VQ-VAE autoencoders have become a *de facto* standard approach to visual tokenization (Van Den Oord et al., 2017; Esser et al., 2021; Kingma & Welling, 2013). They encode the original pixels into a compressed, fixed-size representation, preserving the original signal's structure. This approach is widely applied across different visual domains (Mizrahi et al., 2023; Chang et al., 2022; 2023; Li et al., 2023b; Villegas et al., 2022; Hu et al., 2023), with (Wang & Jiang, 2024; Lu et al., 2025; Ma et al., 2025) developing unified tokenizers across mul-

[1] By "video models" we refer to a broad class of models that have video as an output (potentially, represented by a latent space), including, e.g., text-to-video, image-to-video, and world models.

[2] Up to 256× fewer bytes per frame compared to most standard discrete video tokenizers (Tang et al., 2024; Agarwal, 2025; Yu et al., 2023a).

tiple modalities. Ge et al. (2022); Yu et al. (2023a;b); Yan et al. (2021); Li et al. (2024b); Tang et al. (2024) adopt similar techniques, compressing videos both spatially and temporally into a spatiotemporal 3D token grid. In contrast, `VideoFlexTok` resamples the original video signal into a variable-length coarse-to-fine sequence of tokens not tied to local patches. This enables representing the underlying signal at varying levels of detail depending on its inherent complexity and downstream needs.

1D and semantic tokenization. More recent works rethink the standard VAE-based grid tokenization by resampling images and videos into compact *1D* sequences (Yu et al., 2024; Wang et al., 2024a), enabling *flexible-length* tokenization (Bachmann et al., 2025; Duggal et al., 2024; Yan et al., 2024; Miwa et al., 2025; Wang et al., 2024b; Wen et al., 2025), or introducing a *semantic bias* during tokenization training (Hu et al., 2023; Ma et al., 2025; Lu et al., 2025; Yu et al., 2025). `VideoFlexTok` adopts resampling and variable-length tokenization from FlexTok (Bachmann et al., 2025), and DINO-based semantic bias (Yu et al., 2025), combining and tailoring these components to the video domain. Unlike ElasticTok (Yan et al., 2024), our tokenizer achieves a much higher compression rate, and we demonstrate its benefits beyond compression, showing, for example, its compute-efficiency in downstream video generation tasks.

Video modeling in abstract spaces. Different from modeling in reconstruction-based spaces, Assran et al. (2025); Zhou et al. (2024); Walker et al. (2025) explore learning temporal dynamics in more abstract spaces, e.g., semantic features of pre-trained vision models (Oquab et al., 2023; Radford et al., 2021; Zhai et al., 2023) or down-sampled VAE latents (Jin et al., 2025). More recently, Yin et al. (2025); Li et al. (2024b) show that a hierarchical approach of predicting first abstract tokens and then decoding them into the pixel space leads to more efficient image and video modeling. Our work contributes to this area by developing a distinct flexible representation with a coarse-to-fine structure that can vary its compactness level, adapting to specific downstream needs, unlike commonly adopted fixed-sized representations from pre-trained off-the-shelf models.

3. Method

We start by describing the properties we want to incorporate into `VideoFlexTok`, motivating the particular design choices we make. In the subsequent sections, we describe our method in more detail (see Figure 3 for an overview).

`VideoFlexTok` is an autoencoder. It *encodes a video into a flexible-length sequence of tokens*[3] *representing it in a coarse-to-fine manner.* We follow (Bachmann et al., 2025) and use encoder with register tokens (Darcet et al., 2023; Yu et al., 2024) that resamples the original spatiotemporal video grid into a two-dimensional structure, where the first dimension corresponds to time and the second to the coarse-to-fine structure. To induce the coarse-to-fine hierarchy along the second dimension, we use nested dropout (Kusupati et al., 2022; Wang et al., 2024b; Bachmann et al., 2025), which drops a random number of register tokens from the end. While this alone induces the structure, reconstruction-focused objectives tend to prioritize low-level details (Van Den Oord et al., 2017; Esser et al., 2021; Rombach et al., 2022), preventing first tokens from capturing semantically meaningful information. We, therefore, use a semantic bias by distilling features from a pre-trained vision encoder (Hu et al., 2023; Bachmann et al., 2025; Ma et al., 2025). Note that *no direct supervision is applied as to what information should be encoded in each level of hierarchy, which is purely emergent* through the variable compression mechanism. In addition, since we use DINOv2 (Oquab et al., 2023) as the vision encoder, which is trained in a self-supervised way, no semantic label supervision is applied through it. This first property enables representing videos with a varying levels of detail, which can be adapted to specific downstream needs (see Figures 1 and 2).

Second, the decoder converts any number of tokens into a realistic, plausible video. To enable this, we train the decoder as a generative flow-based model (Bachmann et al., 2025; Ge et al., 2023). In our evaluations, we show that this property allows us to train a downstream conditional generative model, e.g., text-to-video, to produce shorter token sequences that focus on the most relevant information and reusing the decoder to map them into the pixel space, considerably reducing training cost while still producing realistic samples that align with the given conditioning.

3.1. Encoding videos into flexible-length representation

Given the 3D spatiotemporal video VAE latents (Tang et al., 2024) $p \in \mathbb{R}^{T \times HW \times D}$ (after flattening along the spatial dimension), and learnable register tokens $r \in \mathbb{R}^{T \times K \times D}$ (Darcet et al., 2023), we construct the input sequence by interleaving them along the temporal dimension $[p_1, r_1, \ldots, p_T, r_T]$. We operate in the VAE latent space mainly to reduce the cost of training the flow decoder (Rombach et al., 2022). We refer to p_t as a latent frame and to K as *the maximum number of tokens per latent frame*. We then pass this sequence through the encoder with the time-causal attention mask, where the tokens $\{p_t, r_t\}$ for each latent frame can only attend to the past latent frames $\{p_i, p_i\}_{i<t}$. In contrast to (Wang et al., 2024a), which uses full self-attention and represents videos as a flat 1D sequence, our design preserves the time-causal structure of the original signal. This design enables streaming-compatible tokenization, where frames are processed sequentially without requiring access to future frames, and improves downstream gener-

[3] We use tokens and representations interchangeably.

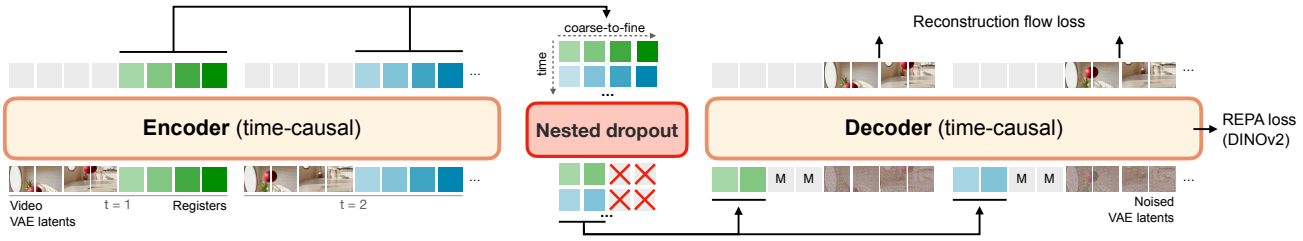

Figure 3. **`VideoFlexTok` overview.** *The encoder* takes the spatiotemporal VAE video latents, interleaves them with learnable register tokens across the time dimension, and passes them through the Transformer with a time-causal attention pattern. This results in a *2D representation with the temporal and coarse-to-fine dimensions*. *Nested dropout* randomly drops $k < K$ last register tokens along the 2nd dimension to induce the ordered coarse-to-fine structure. *The decoder* is a flow-based generative model that temporally interleaves masked register tokens with noisy VAE latents and passes them through a time-causal Transformer. *The losses* are: 1) the rectified flow reconstructive loss that predicts clean patches from their noised version and tokens, and 2) the representation alignment loss (Yu et al., 2025) between the decoder and DINOv2 (Oquab et al., 2023) features, which distills semantic information into `VideoFlexTok` tokens.

ation performance, as we find in Section 4.5. In addition, we follow (Bachmann et al., 2025) and use a causal self-attention mask within the register tokens, and we did not find alternative masking patterns to improve performance.

Since our downstream architecture is an autoregressive GPT-like Transformer with cross-entropy loss, we apply FSQ (Mentzer et al., 2023) quantization (64000 codebook size) to the register tokens' output for discretization and use it as the video representation, denoted as $\hat{r}$. Finally, before decoding, we apply nested dropout (Kusupati et al., 2022) to their second dimension. Specifically, we randomly choose $1 \leq k \leq K$ and mask the last $K - k$ tokens for each $\hat{r}_t$.

3.2. Generative decoder with semantic bias loss

After the encoder, we interleave the masked registers $\hat{r}$ with the noised VAE latents $\tilde{x} = \alpha \cdot \epsilon + (1-\alpha) \cdot x$ along the time dimension $[\hat{r}_1, \tilde{x}_1, \ldots, \hat{r}_T, \tilde{x}_T]$, pass them through the DiT decoder (Peebles & Xie, 2023), and apply a rectified-flow loss (Liu et al., 2022). In addition to the flow objective, we add a semantic bias loss in the form of REPA (Yu et al., 2025), which adds a small readout head to an (early) layer of the decoder to predicts self-supervised DINOv2 features and applies a cosine-similarity loss. While originally proposed to improve the diffusion decoder's training efficiency, previous work (Bachmann et al., 2025; Wen et al., 2025), as well as our early experiments (see Section C.1), suggest that it leads to more semantically aware representations in the encoder-decoder architecture. Our final objective, therefore, is as follows: $\mathcal{L}(\theta) = \mathcal{L}_{\text{Flow}} + \lambda \cdot \mathcal{L}_{\text{REPA}}$, where θ includes the parameters of the encoder, decoder, the REPA head, and the register token queries for the encoder.

In addition to the REPA objective, we use time-causal attention mask in our decoder, which, together with the time-causal encoder and nested dropout, results in a predictive self-supervised objective. Indeed, each $\hat{r}_t$ needs not only to encode information useful for reconstructing the current frame p_t but also for *predicting* all future frames $\{p_i\}_{i>t}$, which was found to be an effective self-supervised objective (Tong et al., 2022; Bardes et al., 2024; Rajasegaran et al., 2025). In Sections C.2 and 4.5, we also find that this design choice leads to better downstream generative performance compared to full attention decoder.

3.3. Downstream autoregressive generation

We evaluate `VideoFlexTok` on conditional video generation tasks. Specifically, we follow (Yu et al., 2023b; Wang et al., 2024a; Bachmann et al., 2025; Agarwal, 2025) and train a GPT-like conditional autoregressive Transformer for class-to-video (C2V) and text-to-video (T2V) tasks. Importantly, we not only focus on the overall fidelity of the generated samples, commonly measured using FVD (Unterthiner et al., 2018), but also measure how well the model solves the conditioning task (as described in Section 4.1), highlighting the semantic properties of our tokenizer.

3.4. Long video tokenization and generation

How can we extend a tokenizer to handle videos longer than it was trained on? One of the main challenges is preserving temporal consistency as we decode subsequent video chunks. This challenge is especially pronounced when decoding from a few `VideoFlexTok` tokens. In this case, the decoder needs to "fill-in" details not present in the tokens, which should be preserved over time as we decode future chunks. Similar to (Li et al., 2024b), we use the following approach. First, we split a video into fixed-length chunks with n overlapping frames and encode each independently. During decoding, we decode the first chunk as usual, and for subsequent chunks, we condition the flow decoder on the last n generated frames. This allows us to preserve temporal consistency in the decoded video even when reconstructing from a few tokens (see Figures 1, 4 and 8).

During downstream generative modeling, this `VideoFlexTok`'s design and aforementioned properties enable the downstream AR transformer to model longer-range temporal dependencies without extensively

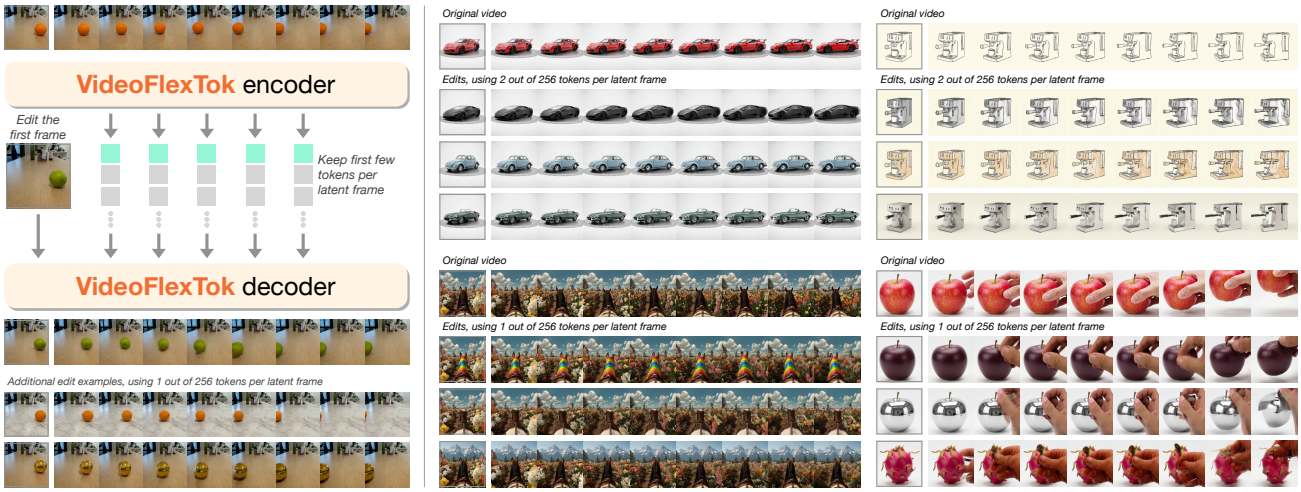

Figure 4. **Probing the first `VideoFlexTok` tokens.** We design the following probing experiment to analyze the information contained in the first `VideoFlexTok` tokens. Given a source video, we keep only one or two tokens per latent frame and make an isolated change to its first frame, e.g., changing an orange to an apple, using Nano Banana (Google, 2025). We then condition the decoder on both the original tokens and the new edited frame for reconstruction. We find that, in most cases, `VideoFlexTok` preserves the motion pattern from the original video and visual appearance from the edited frame throughout the reconstructed video, *suggesting that the first tokens primarily capture the motion information.*

increasing its context length. Specifically, we can now train the AR model to predict only the first few tokens per latent frame capturing the most essential information and use `VideoFlexTok` to decode it back into a consistent video. In Section 4.4, we explore this design and provide a qualitative example. We use 32 tokens per frame, allowing the AR model to generate a 10-second video using only 672 tokens while still expressing the conditioning well (see Figures 1 and 8). For calibration, an off-the-shelf tokenizer (Yu et al., 2023b; Agarwal, 2025; Wang & Jiang, 2024) would require 5376 tokens, extensively increasing both the training and inference cost. This essentially enables to keep longer videos in the context of the AR transformer using a lower token budget.

4. Experiments

4.1. Implementation details

`VideoFlexTok` architecture. We train our tokenizer in the VAE latent space to reduce the computational cost of training the generative flow decoder (Rombach et al., 2022). We use VidTok 3D VAE (Tang et al., 2024), that maps the original video of shape $(T+1) \times H \times W \times 3$ to $(\frac{T}{f_t}+1) \times \frac{H}{f_h} \times \frac{W}{f_w} \times C$. We use the version with $C = 16$ channels and $f = (4, 8, 8)$, which we found to provide a good balance between compactness and reconstruction fidelity. We use a total of 256 registers per each (latent) frame. We parametrize the encoder and decoder Transformer shapes as $\text{depth} = d$, $\text{width} = 64d$, $\text{num\_heads} = d$ following (Tian et al., 2024). For Kinetics-600, we train the tokenizer with $d_{\text{enc}} = d_{\text{dec}} = 18$. For Panda, we use $d_{\text{enc}} = 18$, $d_{\text{dec}} = 28$ and apply additional $[1, 2, 2]$ patchification of the VAE latents to reduce the sequence length.

AR model training. We employ a LLaMa-like Transformer (Touvron et al., 2023; Sun et al., 2024) as our autoregressive generative model. For class-conditional generation, we add a class embedding to the `[SOS]` token. For text-to-video generation, we use T5 (Raffel et al., 2020) as the text encoder and add cross-attention layers to the autoregressive Transformer following (Sun et al., 2024; Kondratyuk et al., 2023). We use the time-first order for `VideoFlexTok`, i.e., we predict the first token for each timestep, then the second and so on, which provides the best overall performance (see Section C.3). We use standard raster-scan order for the 3D grid baseline.

To study AR scaling on `VideoFlexTok` tokens, we follow a compute-aware procedure inspired by Chinchilla-style optimal training (Hoffmann et al., 2022). For our data-rich text-to-video settings, we scale the model size N and training tokens D jointly using the heuristic $D \approx 20N$, and increase the batch size sublinearly with D following a square-root power law to remain within the optimal training regime (Zhang et al., 2024). This defines a FLOPs sweep from 1.6×10^{20} to 5×10^{21} for models from 400M to 5.2B parameters. In the data-limited class-to-video setting, we, instead, fix either D or N and vary the other parameter.

Datasets. For class-to-video generation, we use videos from the Kinetics-600 (Kay et al., 2017; Carreira et al., 2018) dataset at a resolution of 128×128. For text-to-video generation, we use a subset of Panda70M (Chen et al., 2024b) with detailed synthetic captions generated following (Chen et al., 2024a), and use a resolution of 256×256. For both

Figure 5. **Flexible-length autoregressive text-to-video generation.** A text-to-video generative model using `VideoFlexTok` tokens can generate token sequences of varying length for a given conditioning. All token budgets lead to plausible generations, with 2-4 tokens/frame capturing the overall scene details and motion described in the text conditioning well (e.g., the balloon movement), while generating more tokens can express more fine-grained details (e.g., the number of floors).

datasets, we extract 17-frame 4-second clips during training of both the tokenizer and autoregressive models, except in Section 4.4, where we use 81-frame 10-second videos for the autoregressive model training. Note that in both cases, we model substantially longer videos than the more standard ∼0.5 seconds (Wang et al., 2024a; Yu et al., 2023a).

Evaluation metrics. We focus our evaluations on two aspects, fidelity and conditioning alignment. We use Frechét Video Distance (FVD) (Unterthiner et al., 2018) for both generation (gFVD) and reconstruction (rFVD) fidelity. We follow VBench (Huang et al., 2024) and use a UMT-L (Li et al., 2023a) model finetuned for Kinetics-600 classification to measure class-video alignment (using the first 16 out of 17 frames), and ViCLIP-InternVid-10M-FLT (Wang et al., 2023) to measure text-video alignment (subsampling 8 out of 17 frames using a temporal stride of 2). In both cases, we interpolate the videos to 224x224 resolution.

We provide more implementation details in Section F.

4.2. Flexible-length tokenization and generation

Tokenization. Figures 1 and 2 show that `VideoFlexTok` can represent videos in a coarse-to-fine manner with the flow decoder producing plausible generations based on any number of tokens. Importantly, we find that first tokens in the hierarchy capture semantically-meaningful information, such as object type, their motion and overall scene geometry, while abstracting away more nuanced details, such as color information. In Figure 4, we further probe what type of information is encoded in the first tokens, by conditioning the decoder on 1 or 2 tokens per latent frame from a source video and an edited first frame of the same video. We find that `VideoFlexTok` decoder preserves the visual edits made to the first frame and applies the motion from the source video, e.g., by transforming a rolling orange into a rolling apple. This suggests that the first tokens primarily capture the motion information.

Generation. Training an AR model on `VideoFlexTok` tokens naturally leads to a coarse-to-fine autoregressive generation order. Figures 5 and 7 demonstrate how the text-to-video generative model expresses the text conditioning with better precision as generates more tokens. Interestingly, Figure 7 suggests a trade-off in terms of the fidelity between the amount of information generated by the AR model and the flow decoder, suggesting that balancing compute between the AR and flow generative models might be a more efficient strategy.

These results suggest that we can train smaller generative models with fewer training steps by representing videos with shorter sequences, thereby reducing downstream computation costs while achieving similar performance. We demonstrate this quantitatively in the next section.

4.3. Downstream efficiency via flexible compactness

Class-to-video: model size and training time efficiency. Figure 6 shows that adjusting the number of generated `VideoFlexTok` tokens to the specific downstream needs leads to substantial efficiency gains. Specifically, we find that we can train a 5-10× smaller AR model or use 5-10× fewer training tokens (not to be confused with the sequence length per video) to achieve comparable or better performance than the 3D grid tokenizer, which always needs all tokens to be generated. In addition, we find from the mid-

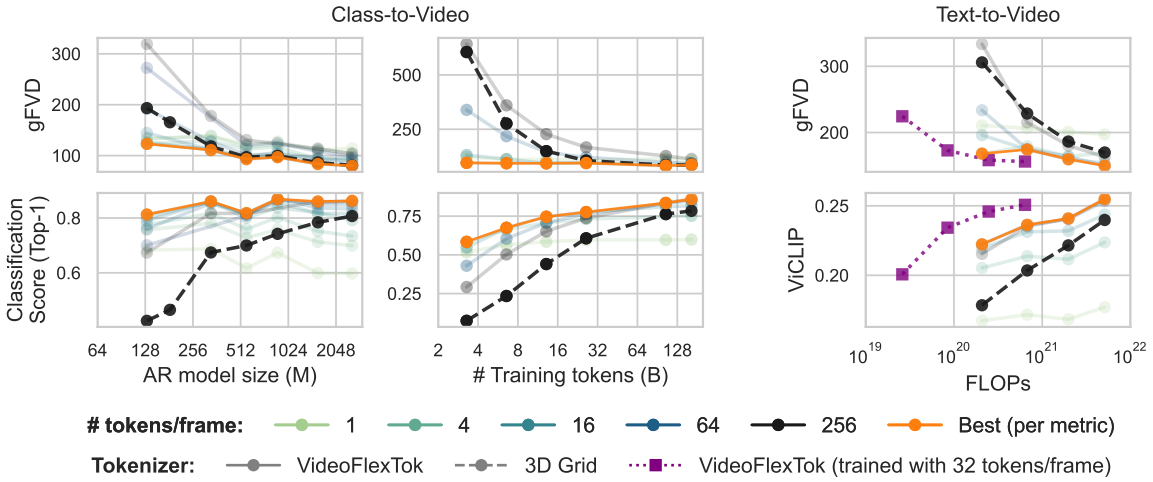

Figure 6. **Compute-efficient AR training with `VideoFlexTok`.** We show how the fidelity (top) and alignment (bottom) metrics change across three complementary scaling axes. **Scaling the model size (left).** We show how the fidelity (top, gFVD) and alignment (bottom, Classification Score) metrics change as we scale the size of the class-to-video autoregressive model. Using `VideoFlexTok` maintains good fidelity across a wider range of model sizes while solving the conditioning task well, achieving a much higher alignment score with smaller models. *This implies that we can train much smaller models to solve the class-to-video downstream task.* **Scaling the number of training tokens (middle).** We show how the fidelity (top, gFVD) and alignment (bottom, Classification Score) metrics evolve during training of the class-to-video downstream model. We use the 1.3B AR model size for this experiment. Using `VideoFlexTok` enables having good reconstructions throughout the whole training, and *achieves similar or better alignment using 5–10 times fewer training tokens than the 3D Grid tokenizer.* **Text-to-video FLOPs efficiency (right).** We show how the fidelity (top, gFVD) and alignment (bottom, ViCLIP Score) metrics change as we scale both the model size and the number of training tokens in the compute-optimal way. We vary the model size from 400M to 5.2B with the estimated ratio of $D/N = 20$ (Hoffmann et al., 2022). In addition to models trained on full-length `VideoFlexTok` token sequences, we train a model on shorter 32-token sequences per latent frame while keeping the number of steps the same, resulting in much lower computational cost (purple line). *Overall, we find that autoregressive generative modeling over `VideoFlexTok` tokens can achieve similar performance using an order of magnitude less compute.*

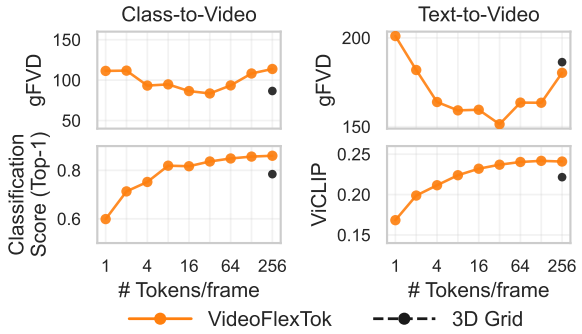

Figure 7. **Flexible-length generation.** We measure fidelity (top, gFVD), and alignment (bottom, classification score and ViCLIP similarity, see Section 4.1) for `VideoFlexTok` and 3D grid tokenizers on class-to-video (left) and text-to-video (right) tasks. Using much fewer tokens, `VideoFlexTok` maintains fidelity comparable to or better than the 3D tokenizer, while achieving higher alignment, i.e., better solving the corresponding conditional generation task.

dle plot, showing the metrics' progress as we increase the number of training tokens, that it is not necessary to train different AR models for each specific sequence length. Indeed, in this experiment, we use full sequences during training and find that the model can generate shorter-sequences well already early in training. Naturally, alignment performance of short sequences (1-4 tokens/frame) eventually saturates, while the performance of longer sequences (64-256) steadily increases. This allows a practitioner to easily achieve performance better than the fixed 3D grid tokenizer across any compute regime, without retraining the tokenizer or the AR model.

Text-to-video: FLOPs efficiency. Since our text-to-video dataset is orders of magnitude larger, we scale both the model size and the number of training tokens in a compute-optimal-inspired way as described in Section 4.1. Similar to the class-to-video results, we find that using `VideoFlexTok` and adjusting the number of the AR-generated tokens achieves performance comparable to the 3D grid counterpart with *an order of magnitude less compute and outperforms it in our largest tested compute regime.* In addition, we train a series of models using shorter sequences (32 tokens/frame) during training, which further significantly reduces the training cost while still achieving comparable performance.

While we focus on analyzing training compute scaling, inference cost is another axis of interest. Indeed, generating fewer tokens with the AR model might require more denoising steps with the flow decoder. In Section E, however, we show that this compute allocation leads to better performance across different inference budgets. In addition, methods that reduce the number of denoising steps can further reduce the inference cost of the flow decoder (Salimans

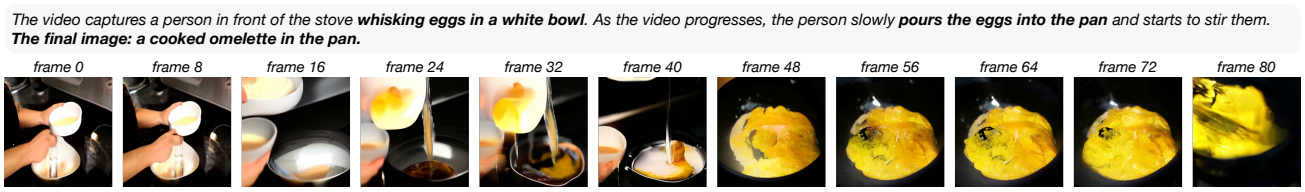

Figure 8. **Long text-to-video generation.** We show an exemplar generation of a 10-second 81-frame video using only 672 tokens (32 tokens per frame).

Table 1. **System-level comparison on Kinetics-600 class-to-video generation.** For each tokenizer, we train a 2.2B class-conditional AR model and evaluate their reconstruction (rFVD) and generation quality in terms of fidelity (gFVD) and alignment (Cls. Score). † indicates `VideoFlexTok` results for a sequence of 160 tokens. ‡ uses the resolution of 256, which is downsampled to 128 before computing the metrics.

Tokenizer	# Tokens	rFVD (↓)	gFVD (↓)	Cls. Score (↑)
VidTok FSQ (Tang et al., 2024)	1280	84.1	131.7	0.799
Cosmos-DV (Agarwal, 2025)	1280	220.5	187.6	0.825
Omnitokenizer (Wang & Jiang, 2024)	1280	63.6‡	102.6‡	0.858‡
LARP (Wang et al., 2024a)	1024	42.1	87.5	0.739
`VideoFlexTok`	5-1280	48.7†	80.0†	0.833†

& Ho, 2022; Yin et al., 2024; Lu et al., 2022).

4.4. Long video generation

Finally, we provide an example of how the above efficiency gains can enable longer temporal modeling without substantially increasing the computational cost of training. As also described in Section 3.4, we train a text-to-video model on 10-second 81-frame videos ($\sim$ 5x longer than in the previous experiments). Building on our previous findings, we use 32 tokens per frame, resulting in only 672 tokens per video (for calibration, a comparable 3D grid tokenizer would require 5376 tokens). We train a 3.2B model for $\sim$55B tokens, resulting in $\sim 10^{21}$ total FLOPs (the middle range of our scaling experiments). Figures 1 and 8 show exemplar generations. The model can generate coherent, 10-second videos that generally follow the text conditioning, all without exceeding the computational budget and context length of shorter-length models from Section 4.3.

4.5. Additional Results

This section presents ablations on some design choices. Section C provides more results, including the effects of REPA and decoder attention, and comparisons between AR orders.

Flat 1D vs time-causal 2D registers structure. Table 2 compares our 2D register design choice, which preserves the time-causal structure of the original video signal, with the 1D flat token structure of LARP (Wang et al., 2024a). We find that while 1D tokens lead to better reconstruction quality (rFVD) due to the encoder's full self-attention, their downstream generative performance is worse (gFVD), suggesting these tokens are harder to predict, likely due to a lack of sufficient structure.

Table 2. **1D vs 2D registers structure.**

Register Structure	rFVD (↓)	gFVD (↓)
Flat (1D)	**48.9**	352.1
Time-causal (2D)	69.9	**287.6**

Decoder self-attention. Table 3 compares the full and time-causal decoder self-attention patterns. We find that while full self-attention leads to better reconstruction performance, the time-causal pattern yields better downstream generative performance, suggesting that it induces additional useful structure into the tokens. Section C.2 further shows that it also leads to better alignment with only a few first tokens, suggesting that these tokens better capture semantic information.

Table 3. **Decoder self-attention pattern.**

Decoder Attention	rFVD (↓)	gFVD (↓) 32 Tok
Full	**58.3**	211.5
Time-Causal	80.9	**175.1**

Comparison to off-the-shelf tokenizers. Table 1 compares `VideoFlexTok` to relevant existing video tokenizers (Tang et al., 2024; Wang et al., 2024a; Agarwal, 2025; Wang & Jiang, 2024). For each tokenizer, we train a 2.2B class-to-video AR model for 164B tokens (except LARP, which sees the same number of videos but slightly fewer tokens due to a higher compression rate) and evaluate their reconstruction and generative performance. While using only 160 tokens (6–8$\times$ fewer than others) during inference, `VideoFlexTok` achieves reconstruction quality (rFVD) comparable to LARP and better generation performance in terms of fidelity (gFVD) and alignment (Cls. Score), except Omnitokenizer which achieves higher alignment. These restuls indicates that `VideoFlexTok` is highly competitive under a much tighter token budget

5. Conclusion and Discussion

We introduce `VideoFlexTok`, a tokenizer that represents videos with a flexible-length sequence of tokens structured in a coarse-to-fine manner, allowing to adapt these representations to particular downstream needs. Its generative flow decoder can decode realistic videos from any number of tokens. We demonstrate that this structure leads to more computationally efficient generative modeling and can enable the generation of longer videos without substantially increasing the context length and computational cost, effectively democratizing video generative modeling.

We believe that modeling in more compact and semantically-aware abstract representation spaces like `VideoFlexTok` will enable capturing long-range dependencies from videos more efficiently compared to learning them directly from pixels. The coarse-to-fine structure enables capturing the dependencies at different levels of abstractions. This, in turn, can lead to more efficient and performant visual reasoning models that adaptively decide what level of abstraction to work in.

Acknowledgment

We thank Mingfei Gao, David Mizrahi, Enrico Fini, Philipp Dufter, and Erik Daxberger for their feedback and discussion during the early stages of the project. We also thank Jason Toskov, Rishubh Singh, Kunal Singh, and Ali Garjani for their help in preparing the manuscript. This work was supported under project ID **a08** as part of the Swiss AI Initiative, through a grant from the ETH Domain and computational resources provided by the Swiss National Supercomputing Centre (CSCS) under the Alps infrastructure. This work has received funding from the Swiss State Secretariat for Education, Research and lnnovation (SERI).

References

Agarwal, Niket, A. A. M. B. Y. B. E. B. T. C. P. C. e. a. Cosmos world foundation model platform for physical ai. 2025.

Assran, M., Bardes, A., Fan, D., Garrido, Q., Howes, R., Komeili, M., Muckley, M., Rizvi, A., Roberts, C., Sinha, K., Zholus, A., Arnaud, S., Gejji, A., Martin, A., Hogan, F. R., Dugas, D., Bojanowski, P., Khalidov, V., Labatut, P., Massa, F., Szafraniec, M., Krishnakumar, K., Li, Y., Ma, X., Chandar, S., Meier, F., LeCun, Y., Rabbat, M., Ballas, N., at Meta, F., Québec, M., Institute, A., and Montréal, P. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *ArXiv*, 2025. URL https://api.semanticscholar.org/CorpusID:279306055.

Bachmann, R., Allardice, J., Mizrahi, D., Fini, E., Kar, O. F., Amirloo, E., El-Nouby, A., Zamir, A., and Dehghan, A. Flextok: Resampling images into 1d token sequences of flexible length. In *Forty-second International Conference on Machine Learning*, 2025.

Bardes, A., Garrido, Q., Ponce, J., Chen, X., Rabbat, M., LeCun, Y., Assran, M., and Ballas, N. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

Burgess, N., Milanovic, J., Stephens, N., Monachopoulos, K., and Mansell, D. H. Bfloat16 processing for neural networks. *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pp. 88–91, 2019. URL https://api.semanticscholar.org/CorpusID:204819410.

Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., and Zisserman, A. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018.

Chang, H., Zhang, H., Jiang, L., Liu, C., and Freeman, W. T. Maskgit: Masked generative image transformer. In *CVPR*, 2022.

Chang, H., Zhang, H., Barber, J., Maschinot, A., Lezama, J., Jiang, L., Yang, M.-H., Murphy, K., Freeman, W. T., Rubinstein, M., et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023.

Chen, L., Li, J., Dong, X., Zhang, P., He, C., Wang, J., Zhao, F., and Lin, D. Sharegpt4v: Improving large multi-modal models with better captions. In *European Conference on Computer Vision*, pp. 370–387. Springer, 2024a.

Chen, T.-S., Siarohin, A., Menapace, W., Deyneka, E., Chao, H.-w., Jeon, B. E., Fang, Y., Lee, H.-Y., Ren, J., Yang, M.-H., et al. Panda-70m: Captioning 70m videos with multiple cross-modality teachers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13320–13331, 2024b.

Darcet, T., Oquab, M., Mairal, J., and Bojanowski, P. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.

DeepMind, G. Veo, 2024. URL https://deepmind.google/models/veo/.

Duggal, S., Isola, P., Torralba, A., and Freeman, W. T. Adaptive length image tokenization via recurrent allocation. *arxiv*, 2024.

Esser, P., Rombach, R., and Ommer, B. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Ge, S., Hayes, T., Yang, H., Yin, X., Pang, G., Jacobs, D., Huang, J.-B., and Parikh, D. Long video generation with time-agnostic vqgan and time-sensitive transformer. In *European Conference on Computer Vision*, pp. 102–118. Springer, 2022.

Ge, Y., Ge, Y., Zeng, Z., Wang, X., and Shan, Y. Planting a seed of vision in large language model. *arXiv preprint arXiv:2307.08041*, 2023.

Google. Introducing Gemini 2.5 Flash Image, our state-of-the-art image model- Google Developers Blog, 2025.

URL https://developers.googleblog.com/introducing-gemini-2-5-flash-image/.

HaCohen, Y., Chiprut, N., Brazowski, B., Shalem, D., Moshe, D., Richardson, E., Levin, E., Shiran, G., Zabari, N., Gordon, O., et al. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Hu, A., Russell, L., Yeo, H., Murez, Z., Fedoseev, G., Kendall, A., Shotton, J., and Corrado, G. Gaia-1: A generative world model for autonomous driving. *ArXiv*, 2023.

Huang, Z., He, Y., Yu, J., Zhang, F., Si, C., Jiang, Y., Zhang, Y., Wu, T., Jin, Q., Chanpaisit, N., et al. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818, 2024.

Jin, Y., Sun, Z., Li, N., Xu, K., Xu, K., Jiang, H., Zhuang, N., Huang, Q., Song, Y., MU, Y., and Lin, Z. Pyramidal flow matching for efficient video generative modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=66NzcRQuOq.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kondratyuk, D., Yu, L., Gu, X., Lezama, J., Huang, J., Schindler, G., Hornung, R., Birodkar, V., Yan, J., Chiu, M.-C., et al. Videopoet: A large language model for zero-shot video generation. *arXiv preprint arXiv:2312.14125*, 2023.

Kong, W., Tian, Q., Zhang, Z., Min, R., Dai, Z., Zhou, J., Xiong, J., Li, X., Wu, B., Zhang, J., et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., Howard-Snyder, W., Chen, K., Kakade, S. M., Jain, P., and Farhadi, A. Matryoshka representation learning. In *Neural Information Processing Systems*, 2022.

Li, K., Wang, Y., Li, Y., Wang, Y., He, Y., Wang, L., and Qiao, Y. Unmasked teacher: Towards training-efficient video foundation models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19891–19903, 2023a. URL https://api.semanticscholar.org/CorpusID:257771777.

Li, T., Chang, H., Mishra, S., Zhang, H., Katabi, D., and Krishnan, D. Mage: Masked generative encoder to unify representation learning and image synthesis. In *CVPR*, 2023b.

Li, T., Katabi, D., and He, K. Return of Unconditional Generation: A Self-supervised Representation Generation Method, November 2024a. URL http://arxiv.org/abs/2312.03701. arXiv:2312.03701 [cs].

Li, Z., Hu, S., Liu, S., Zhou, L., Choi, J., Meng, L., Guo, X., Li, J., Ling, H., and Wei, F. Arlon: Boosting diffusion transformers with autoregressive models for long video generation. *ArXiv*, 2024b. URL https://api.semanticscholar.org/CorpusID:273653802.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017. URL https://api.semanticscholar.org/CorpusID:53592270.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Lu, J., Song, L., Xu, M., Ahn, B., Wang, Y., Chen, C., Dehghan, A., and Yang, Y. Atoken: A unified tokenizer for vision. *ArXiv*, 2025.

Ma, C., Jiang, Y., Wu, J., Yang, J., Yu, X., Yuan, Z., Peng, B., and Qi, X. Unitok: A unified tokenizer for visual generation and understanding. *arXiv preprint arXiv:2502.20321*, 2025.

Mentzer, F., Minnen, D., Agustsson, E., and Tschannen, M. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023.

Miwa, K., Sasaki, K., Arai, H., Takahashi, T., and Yamaguchi, Y. One-d-piece: Image tokenizer meets quality-controllable compression. *ArXiv*, abs/2501.10064, 2025. URL https://api.semanticscholar.org/CorpusID:275606373.

Mizrahi, D., Bachmann, R., Kar, O. F., Yeo, T., Gao, M., Dehghan, A., and Zamir, A. 4M: Massively multimodal masked modeling. In *NeurIPS*, 2023.

OpenAI. Sora. https://openai.com/index/sora, 2024. Accessed: 2025-02-14.

OpenAI. Sora 2, 2025. URL https://openai.com/index/sora-2/. Accessed: 2025-02-14.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In *ICLR*, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21 (140):1–67, 2020.

Rajasegaran, J., Radosavovic, I., Ravishankar, R., Gandelsman, Y., Feichtenhofer, C., and Malik, J. An empirical study of autoregressive pre-training from videos. *arXiv preprint arXiv:2501.05453*, 2025.

Ramanujan, V., Tirumala, K., Aghajanyan, A., Zettlemoyer, L., and Farhadi, A. When worse is better: Navigating the compression generation trade-off in visual tokenization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=o8hWyJIgAV.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Shazeer, N. M. Glu variants improve transformer. *ArXiv*, abs/2002.05202, 2020. URL https://api.semanticscholar.org/CorpusID:211096588.

Sun, P., Jiang, Y., Chen, S., Zhang, S., Peng, B., Luo, P., and Yuan, Z. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

Tang, A., He, T., Guo, J., Cheng, X., Song, L., and Bian, J. Vidtok: A versatile and open-source video tokenizer. *arXiv preprint arXiv:2412.13061*, 2024.

Tian, K., Jiang, Y., Yuan, Z., Peng, B., and Wang, L. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

Tong, Z., Song, Y., Wang, J., and Wang, L. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Unterthiner, T., Van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., and Gelly, S. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

Van Den Oord, A., Vinyals, O., et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Villegas, R., Babaeizadeh, M., Kindermans, P.-J., Moraldo, H., Zhang, H., Saffar, M. T., Castro, S., Kunze, J., and Erhan, D. Phenaki: Variable length video generation from open domain textual description. *arXiv preprint arXiv:2210.02399*, 2022.

Walker, J. C., V'elez, P., Cabrera, L. P., Zhou, G., Kabra, R., Doersch, C., Ovsjanikov, M., Carreira, J., and Ginosar, S. Generalist forecasting with frozen video models via latent diffusion. *ArXiv*, abs/2507.13942, 2025. URL https://api.semanticscholar.org/CorpusID:280048244.

Wan, T., Wang, A., Ai, B., Wen, B., Mao, C., Xie, C.-W., Chen, D., Yu, F., Zhao, H., Yang, J., et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Wang, H., Suri, S., Ren, Y., Chen, H., and Shrivastava, A. Larp: Tokenizing videos with a learned autoregressive generative prior. 2024a. URL https://arxiv.org/abs/2410.21264.

Wang, X. and Aitchison, L. How to set adamw's weight decay as you scale model and dataset size. *ArXiv*, abs/2405.13698, 2024. URL https://api.semanticscholar.org/CorpusID:269982015.

Wang, X., Zhou, X., Fathi, A., Darrell, T., and Schmid, C. Visual lexicon: Rich image features in language space. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

Wang, Y., He, Y., Li, Y., Li, K., Yu, J., Ma, X., Li, X., Chen, G., Chen, X., Wang, Y., et al. Internvid: A large-scale video-text dataset for multimodal understanding and generation. *arXiv preprint arXiv:2307.06942*, 2023.

Wang, Junke, Y. J. Z. Y. B. P. Z. W. and Jiang, Y.-G. Omnitokenizer: A joint image-video tokenizer for visual generation. *Advances in Neural Information Processing Systems*, 2024.

Wen, X., Zhao, B., Elezi, I., Deng, J., and Qi, X. "principal components" enable a new language of images. *ArXiv*, abs/2503.08685, 2025. URL https://api.semanticscholar.org/CorpusID:276929175.

Xu, J., Mei, T., Yao, T., and Rui, Y. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5288–5296, 2016.

Yan, W., Zhang, Y., Abbeel, P., and Srinivas, A. Videogpt: Video generation using vq-vae and transformers. *ArXiv*, 2021. URL https://api.semanticscholar.org/CorpusID:233307257.

Yan, W., Zaharia, M., Mnih, V., Abbeel, P., Faust, A., and Liu, H. Elastictok: Adaptive tokenization for image and video. *arXiv preprint*, 2024.

Yang, G., Hu, J. E., Babuschkin, I., Sidor, S., Liu, X., Farhi, D., Ryder, N., Pachocki, J. W., Chen, W., and Gao, J. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *ArXiv*, abs/2203.03466, 2022. URL https://api.semanticscholar.org/CorpusID:247292726.

Yang, Z., Teng, J., Zheng, W., Ding, M., Huang, S., Xu, J., Yang, Y., Hong, W., Zhang, X., Feng, G., et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

Yin, A., Shen, K., Leng, Y., Tan, X., Zhou, X., Li, J., and Tang, S. The best of both worlds: Integrating language models and diffusion models for video generation. *arXiv preprint arXiv:2503.04606*, 2025.

Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W. T., and Park, T. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6613–6623, 2024.

Yu, L., Cheng, Y., Sohn, K., Lezama, J., Zhang, H., Chang, H., Hauptmann, A. G., Yang, M.-H., Hao, Y., Essa, I., et al. Magvit: Masked generative video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10459–10469, 2023a.

Yu, L., Lezama, J., Gundavarapu, N. B., Versari, L., Sohn, K., Minnen, D., Cheng, Y., Birodkar, V., Gupta, A., Gu, X., et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023b.

Yu, Q., Weber, M., Deng, X., Shen, X., Cremers, D., and Chen, L.-C. An image is worth 32 tokens for reconstruction and generation. *Advances in Neural Information Processing Systems*, 37:128940–128966, 2024.

Yu, S., Kwak, S., Jang, H., Jeong, J., Huang, J., Shin, J., and Xie, S. Representation alignment for generation: Training diffusion transformers is easier than you think. In *ICLR*, 2025.

Zhai, X., Mustafa, B., Kolesnikov, A., and Beyer, L. Sigmoid loss for language image pre-training. *ICCV*, 2023.

Zhang, H., Morwani, D., Vyas, N., Wu, J., Zou, D., Ghai, U., Foster, D., and Kakade, S. How does critical batch size scale in pre-training? *arXiv preprint arXiv:2410.21676*, 2024.

Zheng, B., Ma, N., Tong, S., and Xie, S. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025.

Zheng, Z., Peng, X., Yang, T., Shen, C., Li, S., Liu, H., Zhou, Y., Li, T., and You, Y. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024.

Zhou, G., Pan, H., LeCun, Y., and Pinto, L. Dino-wm: World models on pre-trained visual features enable zero-shot planning, 2024. URL https://arxiv.org/abs/2411.04983.

Zhu, Y., Liu, X., and Liu, Q. Slimflow: Training smaller one-step diffusion models with rectified flow. In *European Conference on Computer Vision*, pp. 342–359. Springer, 2024.

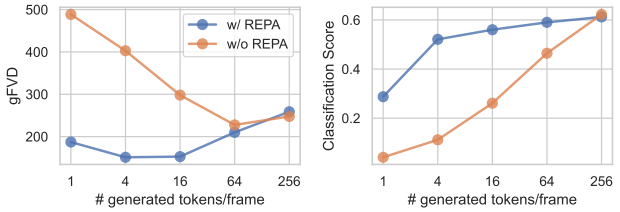

Figure 9. **REPA (Yu et al., 2025) loss ablation.** We compare tokenizers trained with and without the REPA loss on the class-to-video downstream task. We find that REPA inductive bias loss improves both the fidelity of the generated samples and the alignment with the class conditioning.

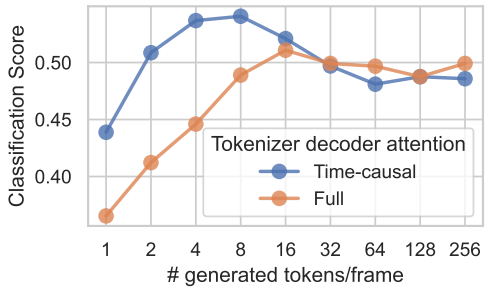

Figure 10. **`VideoFlexTok` decoder attention ablation** We ablate the decoder attention pattern by comparing the alignment score (Classification Score) on the class-to-video generative task. We find that causal attention leads to a higher alignment score with fewer tokens, suggesting the early tokens capture more semantic information in this case (see Section C.2 for discussion).

A. Overview video

We provide an overview video of our submission in `overview.mp4`.

B. Additional qualitative results

In Figures 13 to 16, as well as in the supplementary archive, we provide additional examples of the variable-length video reconstructions by `VideoFlexTok`.

C. Additional ablations

In the ablation experiments, we use the `VideoFlexTok d12-d12` version as described in Table 5 and an autoregressive model with depth 16 and 201M parameters, unless stated otherwise.

C.1. REPA: semantic inductive bias

We ablate the use of the additional REPA loss (Yu et al., 2025). Figure 9 shows that using this loss significantly improves the fidelity and alignment score in the few tokens regime.

C.2. Causal decoder: future prediction pre-training task

As described in Section F, we use time-causal attention in the decoder during encoder training. In Section 4.5 and Table 3, we demonstrate that this design improves downstream generative performance; thus, we adopt it. As discussed in Section 3.2, this causal design, combined with nested dropout, results in a future prediction task, which was found to be a useful self-supervised pre-training objective (Tong et al., 2022; Bardes et al., 2024; Rajasegaran et al., 2025). We hypothesize, therefore, that this design leads to the first register tokens capturing "more" semantic information. We evaluate this hypothesis by comparing the alignment score for class-to-video downstream generation in Figure 10. We find that using a time-causal decoder yields a higher alignment score with the class label for fewer tokens, suggesting that this information is better captured in this case than with the full attention decoder. The lower classification score with more generated tokens can be explained by the fact that we use a relatively small 200M autoregressive model, which results in poor generation quality for the full sequence length (see, e.g., the trend in Figure 7 where we use a 1.3B AR model and causal decoder).

C.3. Autoregressive generation order

In this section, we compare the time-first and the depth-first AR orders. Time-first is the default order we use in our experiments in Section 4.3: we first predict the first token across all latent frames, then the second, and so on. In the depth-first order, we predict all tokens for the first latent frame, then for the second, and so on. Table 4 shows the results. We did not find a significant difference between the two AR orders, and the time-first order allows varying the number of generating tokens, which leads to better performance. While it is possible to train an AR model with depth-first order with fewer tokens per latent frame, it requires training a separate model for each token budget. Another approach could be to design a more flexible generative model that can be controlled to produce a different number of tokens. We leave a more extensive exploration of the best way to predict this 2D, coarse-to-fine $\times$ time `VideoFlexTok` tokens for future work.

D. Hierarchical generation with `VideoFlexTok`

In the main paper, in Section 4.3, we mainly focus on the downstream benefits brought by the flexible compression rate. In this section, we focus on studying the effect of the hierarchical coarse-to-fine autoregressive generation order enabled by `VideoFlexTok`. To this end, Figure 11 compares the performance of `VideoFlexTok` and its 3D-grid controlled counterpart at the same compression rate, i.e.,

Table 4. **Ablating different autoregressive orders over `VideoFlexTok` tokens.** We compare autoregressive orders on the class-to-video downstream task. We use depth-first and time-first orders for the same underlying tokenizer. We find no significant difference when using all tokens. The time-first order allows adjusting the number of tokens during evaluation leading to better performance.

AR order	#Tokens	gFVD (↓)
time-first	1	187.1
	4	**151.1**
	16	152.7
	64	210.1
	256	246.6
depth-first	256	242.6

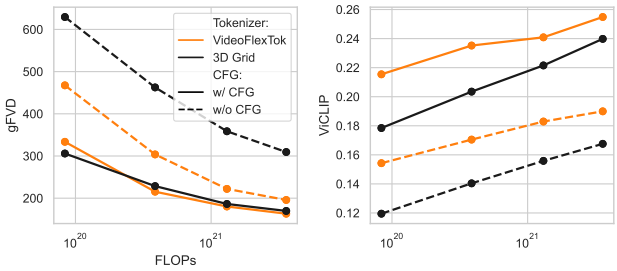

Figure 11. **Hierarchical vs. raster-order generation.** We compare the `VideoFlexTok` and 3D grid tokenizers' performance at the same sequence length (1280 tokens). We find that hierarchical generation with `VideoFlexTok` leads to 1) better alignment (ViCLIP score) and 2) much better fidelity (gFVD) when not using classifier-free guidance.

using the same number of tokens per frame $N = 256$ for both. First, we find that `VideoFlexTok` achieves a better text alignment score across all scales. Second, we find that `VideoFlexTok` is much less reliant on classifier-free guidance for generation fidelity, achieving a much lower gFVD without it. Note that both tokenizers benefit from the REPA inductive bias (see Sections 3.1 and C.1), with the only differences being the token structure and the use of nested dropout. We hypothesize that, as with text or class conditioning, which are crucial for achieving high fidelity (see, e.g., (Li et al., 2024a)) compared to unconditional generation, hierarchical coarse-to-fine generation with `VideoFlexTok` helps split the problem into a sequence of simpler problems, leading to better overall performance.

E. Inference cost analysis

In Section 4.3, we show that using `VideoFlexTok` can drastically reduce the training compute cost and achieve similar performance with smaller models and/or less training. This is mainly achieved by generating fewer tokens with the AR model (though we still see improved alignment even when generating all tokens). This, however, inquires the cost of running the flow decoder for multiple steps to generate the missing details and obtain the final full RGB output (VAE latents in our case). In this section, we study how the performance changes as we scale the inference compute by either sampling more tokens with the AR model or doing more denoising steps with the `VideoFlexTok` decoder for different AR model sizes. We use the `VideoFlexTok d18-d28` tokenizer for this experiment (see Table 5).

Figure 12 shows that for all considered model sizes and inference costs, generating less than 256 tokens per frame (the full sequence) and using the flow decoder achieves a better performance for all inference budgets. We believe that this trend can be further improved by balancing compute between the AR and flow decoder models more carefully. In addition, the flow inference cost can be significantly reduced by using distillation-based approaches (Salimans & Ho, 2022; Yin et al., 2024; Lu et al., 2022; Zhu et al., 2024). We leave this direction to future work.

F. Architecture and training details

F.1. `VideoFlexTok` details

We provide a detailed overview of the architecture and training configuration in Table 5. We follow Bachmann et al. (2025) for our overall design and introduce the following changes, extending it to video sequences.

- We extend 1D registers to 2D by adding the time dimension as described in Section 3.1.
- In addition to causal attention over register tokens within a latent frame, we also introduce a time-causal attention mask in both the encoder and decoder.
- We use a pre-trained VidTok video VAE (Tang et al., 2024) with both temporal and spatial compression.
- As the REPA (Yu et al., 2025) head, we use a Transformer with time-causal attention mimicking the decoder design, which we found to perform better in terms of both reconstruction and downstream generation performance in our early explorations.
- We introduce an additional decoder fine-tuning stage where we keep the encoder frozen and make the following changes. First, we use full attention, which leads to more temporally-consistent reconstructions and better overall fidelity. Note that it is important to freeze the encoder during this stage, as using full attention during training leads to worse performance as we demonstrate in Section 4.5 and Table 3. Second, we introduce a frame-conditioning capability by randomly providing a clean VAE latent for the first latent frame,

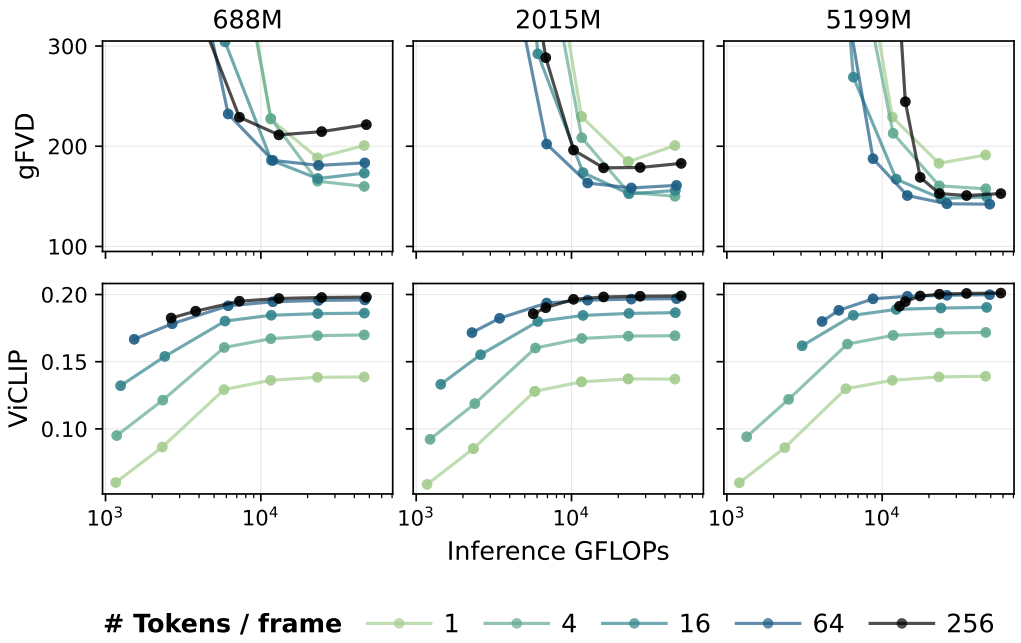

Figure 12. **Text-to-video inference cost analysis.** We compare the inference cost of various configurations of the number of AR-generated tokens and the number of `VideoFlexTok` flow decoder steps. For each AR model size and the number of generated tokens, we perform $\{1, 2, 5, 10, 20, 40\}$ denoising steps and plot the corresponding lines. We find that for all considered AR sizes and inference budgets, the best performance is achieved by generating less than 256 tokens per frame and performing multiple denoising steps.

corresponding to the first frame due to the causal VAE, with probability $p = 0.5$. Similar to (Zheng et al., 2024), we find that even a short fine-tuning is enough to acquire this capability.

F.2. Autoregressive model details

We provide a detailed overview of the architecture and training configuration for the autoregressive model in the class-to-video (Table 6) and text-to-video (Table 7) experiments. The AR models are causal decoder Transformers where the hidden size is tied to the depth via $w = 64d$, the number of attention heads equals the depth d, and the feed-forward layers apply an MLP ratio of 4 relative to the attention dimension. We do not use μP (Yang et al., 2022) for the AR models, instead opting for scaling the learning rate inversely with the model width.

In the class conditioned setting, we mitigate overfitting to the relatively small Kinetics-600 dataset by applying dropout in the FFN, attention, and projection modules of the Transform with probability 0.1 and random cropping with resizing of the videos. All model sizes are trained for the same length 164B tokens.

In the text-to-video setting the training is not as data constrained, so we do not apply dropout in the Transformer. When scaling the AR model we follow compute-optimal training, training the different model sizes using the rule of thumb $D \approx 20N$ (Hoffmann et al., 2022). As we scale the number of training tokens we also scale the batchsize following a square-root relationship and rounding down to the nearest power of 2 (Zhang et al., 2024). We make the assumption for these compute-optimally trained models that the training of a decoder-only model on video data follows similar training token to parameter scaling as for a text-only model. To mitigate potential differences in training models on the two modalities we also train a model which is 4x over-trained relative to the Chinchilla compute optimal value.

G. Reconstruction on MSR-VTT

Table 8 provides a comparison of `VideoFlexTok` for video reconstruction relative to common video tokenizers on the MSR-VTT dataset (Xu et al., 2016), which we use as an out-of-distribution dataset for our tokenizer trained on the large-scale Panda dataset. In these evaluations, we sample 17 frames at 4 FPS, 256 by 256 resolution from 5k samples from the MSR-VTT dataset. The `VideoFlexTok d18-d28` version outperforms the baselines on reconstruction fidelity metrics such as rFID and rFVD with 1280 tokens per video clip. The `VideoFlexTok` is slightly worse on pixel-level reconstruction metrics such as MSE and MAE.

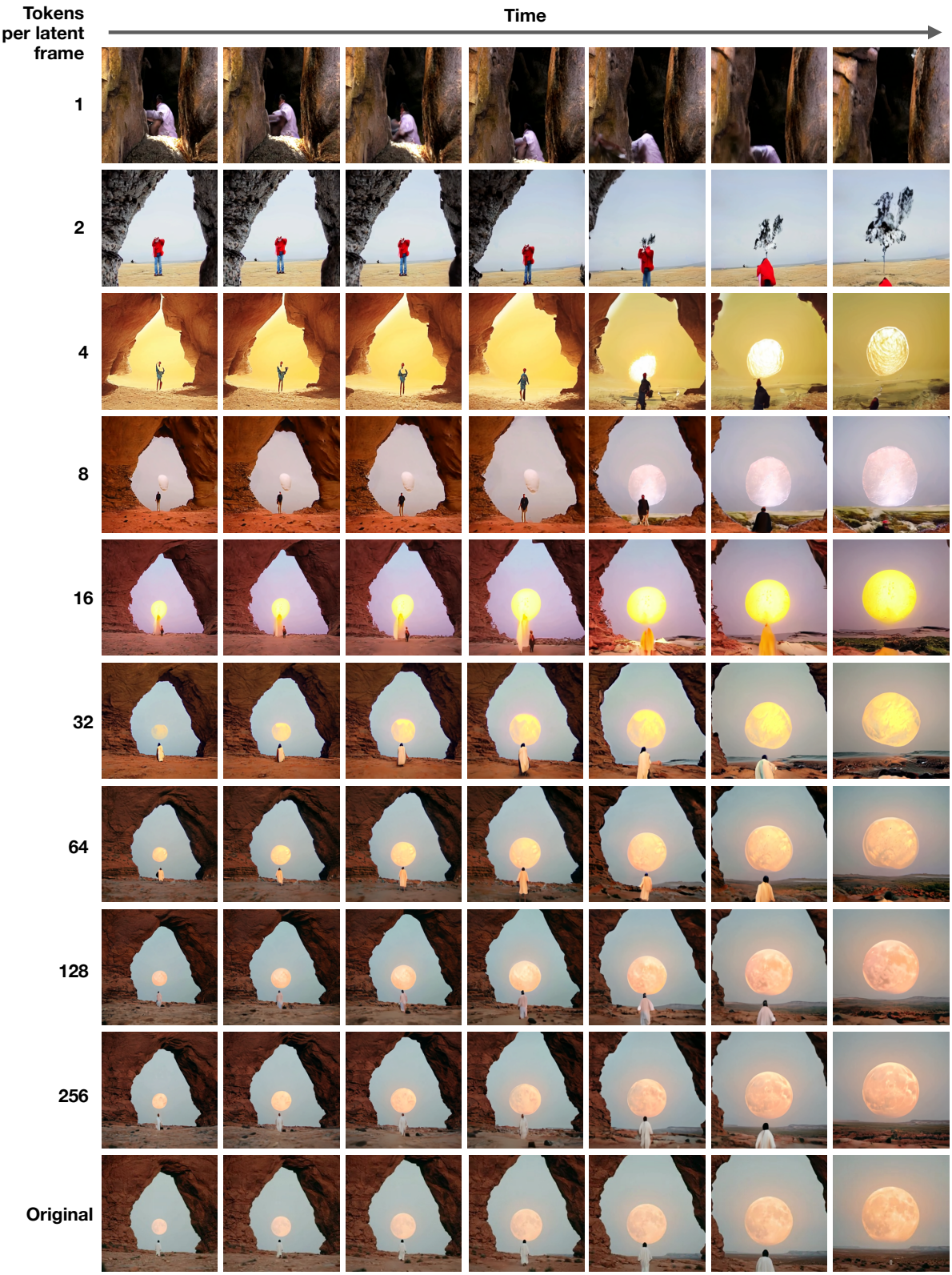

Figure 13. **`VideoFlexTok` reconstruction example**. From top to bottom, each row corresponds to a video reconstructed using 1, 2, 4, . . . , 256 tokens. The last row shows the original video.

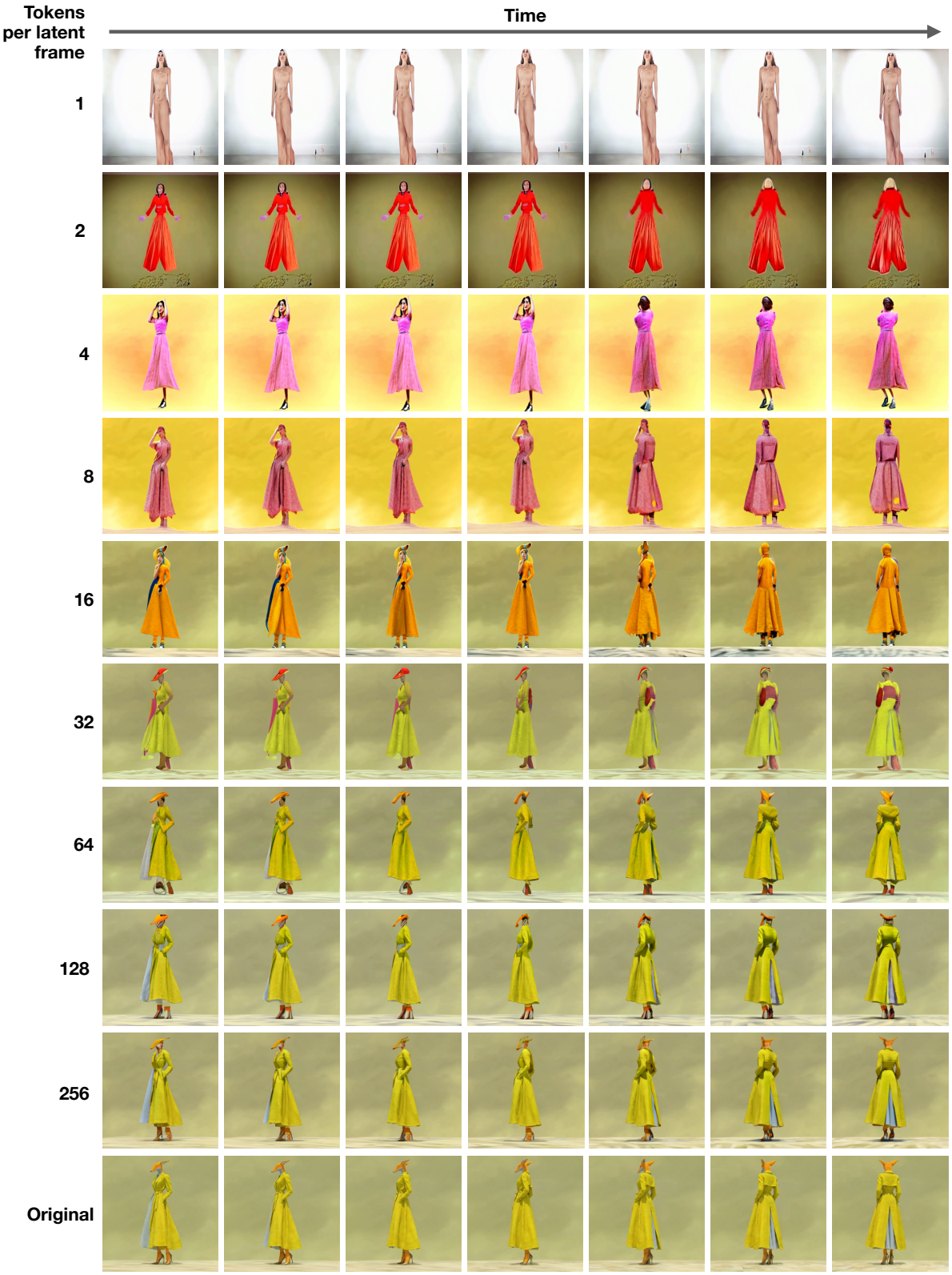

Figure 14. **`VideoFlexTok` reconstruction example**. From top to bottom each row corresponds to a video reconstructed using 1, 2, 4, . . . , 256 tokens. The last row shows the original video.

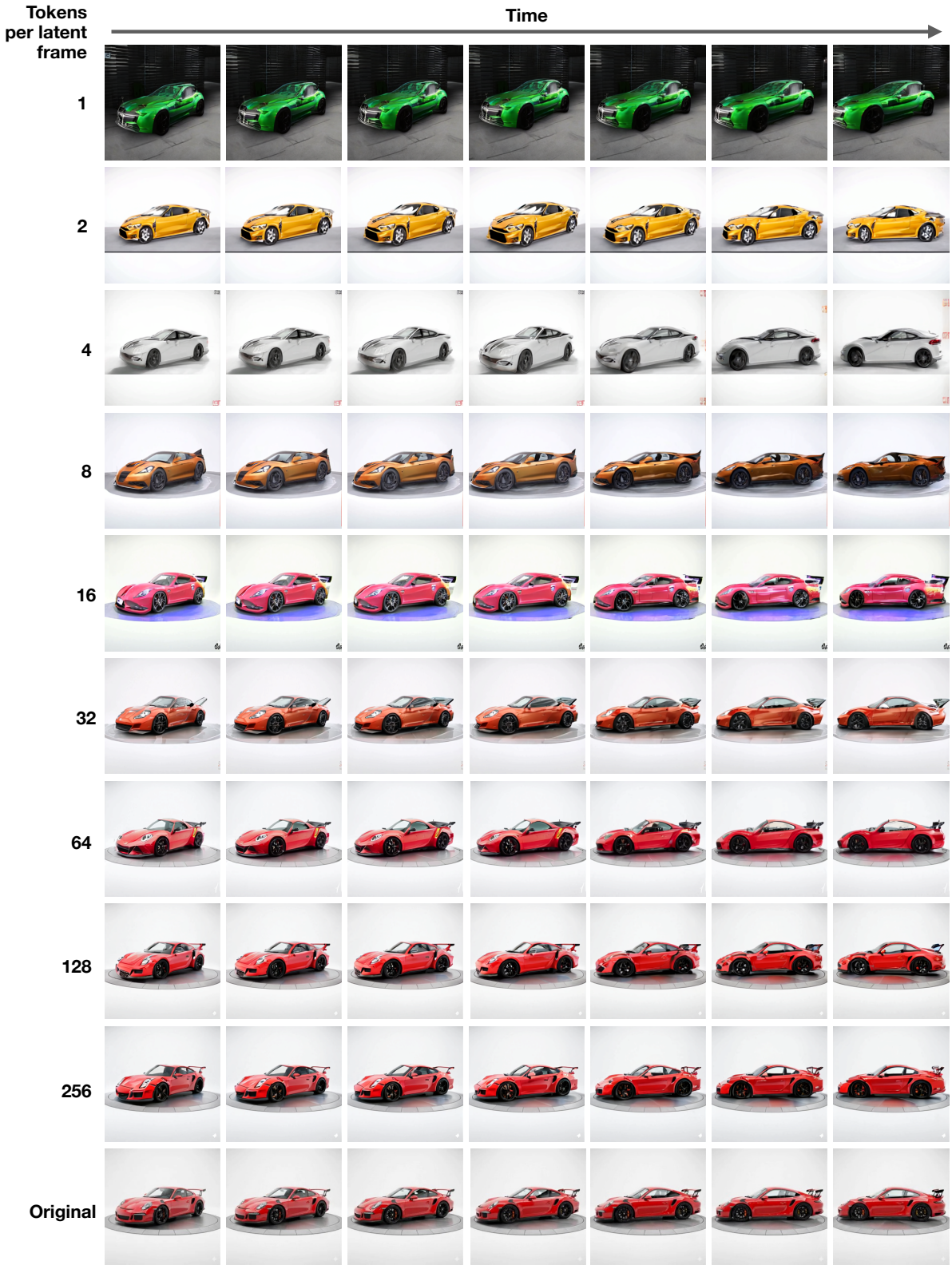

Figure 15. **`VideoFlexTok` reconstruction example**. From top to bottom each row corresponds to a video reconstructed using 1, 2, 4, . . . , 256 tokens. The last row shows the original video.

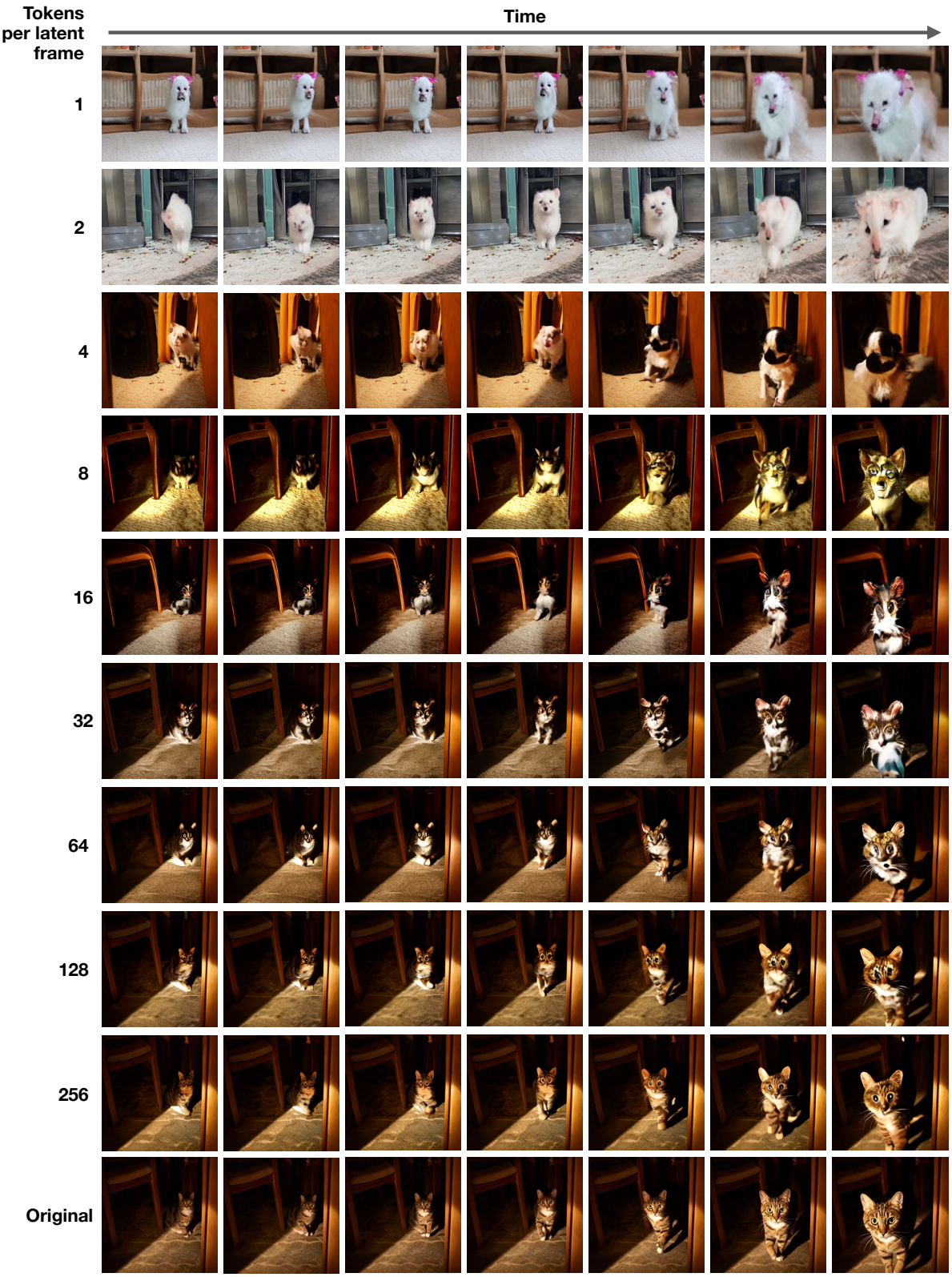

Figure 16. **`VideoFlexTok` reconstruction example**. From top to bottom each row corresponds to a video reconstructed using 1, 2, 4, . . . , 256 tokens. The last row shows the original video.

Table 5. **`VideoFlexTok` training settings.** Model and training configuration for different autoregressive Transformer sizes.

`VideoFlexTok` configuration	`d12-d12` (ablation setting)	`d18-d18`	`d18-d28`
Encoder depth d_{enc}	12	18	18
Decoder depth d_{dec}	12	18	28
Encoder dim. $w_{enc} = 64d_{enc}$	768	1152	1152
Decoder dim. $w_{dec} = 64d_{dec}$	768	1152	1792
Encoder Transformer parameters	84.9M	286.7M	286.7M
Decoder Transformer parameters	84.9M	286.7M	1.1B
Decoder adaLN (Peebles & Xie, 2023) parameters	84.9M	286.7M	1.1B
Max. num. registers K	$5 \times 256 = 1280$		
Encoder attention mask	Time-causal + causal over registers (ref. Section 3.1)		
Decoder attention mask	Time-causal		
Register nested dropout mode	Uniform from $\{1, 2, 4, \ldots, 256\}$ per latent frame (same for all)		
FSQ (Mentzer et al., 2023) levels	$[8, 8, 8, 5, 5, 5]$ (64000 vocab. size)		
VAE (Tang et al., 2024)	`vidtok_kl_causal_488_16chn` (Tang et al., 2024)		
VAE channels	16		
VAE downsampling factor (time×height×width)	$4 \times 8 \times 8$		
Patch size [time, height, width]	`[1, 1, 1]`		`[1, 2, 2]`
REPA (Yu et al., 2025) layer	1		
REPA (Yu et al., 2025) model	DINOv2-L (Oquab et al., 2023)		
REPA (Yu et al., 2025) projection	Time-causal Transformer; depth 2; decoder width		
REPA (Yu et al., 2025) loss weight	1.0		
Training length (n tokens)	100B	400B	400B
Warmup length (n tokens)	4B		
Warmup learning rate	`1e-6`		
Learning rate schedule	Cosine decay		
Optimizer	AdamW		
Opt. momentum	$\beta_1, \beta_2 = 0.9, 0.99$		
Learning rate η	`1.124e-3`		
Batch size, samples	512		10240
μP (Yang et al., 2022) base dim.	256		
Gradient clipping norm	1.0		
Decoder full-attention fine-tuning (n tokens)	-	-	400B
Dataset	Kinetics-600		Panda (30M subset)
Video resolution	`17x128x128`		`17x256x256`
Data type	`bfloat16` (Burgess et al., 2019)		

Table 6. **Class-conditioned AR training settings.** Model and training configuration for different autoregressive Transformer sizes.

Configuration	**AR 49M**	**AR 85M**	**AR 201M**	**AR 393M**	**AR 679M**	**AR 1.33B**	**AR 2.29B**
Num. non-embedding parameters	49M	85M	201M	393M	679M	1.33B	2.29B
Decoder depth d_{dec}	10	12	16	20	24	30	36
Decoder dim. w_{dec}	640	768	1024	1280	1536	1920	2304
Cross-attention dim.	n/a						
MLP ratio	4						
Max. sequence length	1280						
Attention mask	Causal						
Vocab size	64,000						
Feedforward activation	SwiGLU (Shazeer, 2020)						
Positional encoding	Learned embedding						
Conditioning dropout prob.	0.1						
FFN, attn. and proj. dropout prob.	0.1						
Training length (n tokens)	164B						
Warmup length (n tokens)	$\approx$4.1B (2.5% of training tokens)						
Initial warmup learning rate	$\eta \times 1e$-3						
Learning-rate schedule	Linear warmup + cosine decay						
Optimizer	AdamW (Loshchilov & Hutter, 2017)						
Opt. momentum	$\beta_1, \beta_2 = 0.9, 0.95$						
Learning rate η	1.60e-3	1.33e-3	1.00e-3	8.00e-4	6.67e-4	5.33e-4	4.4e-4
Final learning rate	$\eta \times 1e$-2						
Batch size	512						
μP (Yang et al., 2022) base dim.	n/a						
Weight decay	0.05						
Weight-decay timescale τ_{iter} (Wang & Aitchison, 2024)	n/a						
Gradient clipping norm	1.0						
Dataset	Kinetics-600 (Carreira et al., 2018)						
Video resolution	`17x128x128`						
Augmentations	`RandomResizedCrop`						
Data type	`bfloat16` (Burgess et al., 2019)						

Table 7. **Text-conditioned AR training settings.** Model and training configuration for different autoregressive Transformer sizes.

Configuration	AR 400M	AR 1.1B	AR 2.0B	AR 5.2B	AR 5.2B (4×C)
Num. total parameters	400M	110M	2.02B	5.20B	5.20B
Decoder depth d_{dec}	12	20	30	42	42
Decoder dim. w_{dec}	768	1280	1920	2688	2688
Cross-attention dim.	12	20	30	42	42
MLP ratio	4				
Max. sequence length	1280				
Attention mask	Causal				
Vocab size	64,000				
Feedforward activation	SwiGLU (Shazeer, 2020)				
Positional encoding	Learned embedding				
Training length (n tokens)	3.25B	12B	38B	101B	402B
Warmup length (n tokens)	2.5% of training tokens				
Initial warmup learning rate	1e-3 $\times$ η				
Learning rate schedule	Cosine decay				
Optimizer	AdamW (Loshchilov & Hutter, 2017)				
Opt. momentum	$\beta_1, \beta_2 = 0.9, 0.95$				
Learning rate η	1.33e-3	8.00e-4	5.33e-4	3.79e-4	3.79e-4
Final learning rate	$\eta \times$ 1e-2				
Batch size	128	256	512	512	512
μP (Yang et al., 2022) base dim.	n/a				
Weight decay	0.05				
Weight decay timescale τ_{iter} (Wang & Aitchison, 2024)	n/a				
Gradient clipping norm	1.0				
Dataset	Panda-70M (30M subset)				
Video resolution	`17×256×256`				
Augmentations	`RandomResizedCrop`				
Data type	`bfloat16` (Burgess et al., 2019)				

Table 8. **Reconstruction metrics on MSR-VTT.** Evaluation is performed on 5k MSR-VTT videos (17 frames, 4 FPS, 256×256). All tokenizers use the same decoder architecture. † indicates `VideoFlexTok` results using 1280 tokens per video. ‡ marks results obtained from models evaluated at 128×128 input resolution.

Tokenizer	# Tok.	MAE (↓)	MSE (↓)	PSNR (↑)	LPIPS (↓)	rFID (↓)	rFVD (↓)
VidTok FSQ (Tang et al., 2024)	1280	0.0276	0.00270	25.64	0.0967	5.26	35.47
LARP (**?**)	1024	0.0476‡	0.00760‡	21.21‡	**0.1111**‡	5.50‡	28.33‡
Omnitokenizer (Wang & Jiang, 2024)	1280	**0.0206**	**0.00100**	**30.01**	0.1199	7.43	38.67
Cosmos-DV (Agarwal, 2025)	1280	0.0287	0.00284	25.46	0.122	8.51	47.20
`VideoFlexTok d18-d28`	5–1280	0.0607†	0.01079†	19.67†	0.18545†	**4.97**†	**20.53**†